20 September 2025 10:18

Lexographical Sorting
Lexographical Counting

Print numbers till 1000 . => for(int i=0; i<=1000; i++) {
SOP(i);
}

Dictionary order

akash > aastha

k > a

Print numbers from 0 to 1000 { Lexographical }

55 > 7 => Integer

"55" < "7" => Strings

10 < 101 => Integer

"10" < "101" => String

extra character

"11" > "100"

100 < 13

ashu < ashutosh

extra character

```
public class Main {
    public static void main(String args[]) {
        int i = 0;
        for(;;){
            System.out.println(i);
            if(i>7){
                break;
            }
            i++;
        }

    }
}
```

Lexographical numbers

n = 13 => 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 ✕

n = 13

0 1
10 11 12 13 ~~14~~
~~100~~
2 3 4 5 6 7 8 9

n = 1000

0 1 101 11 12 13 14 15 16 17 18 19
10 102 110 120 130
100 103 111 121 131
1000 112 122
109
119 129 139 199

2 3 — — — — 9
20 30 90
200 300 900
201 301 901
202
209 309 909

1 < 20

print numbers from 0 to 30

0
1
10 11 12 13 14 15 16 17 18 19
~~100~~ ~~110~~

Day 18 Page 1

1

1 0 11 12 13 14 15 16 17 18 19

~~100~~ ~~110~~

2

2 0 21 22 23 - - - 29

~~200~~

3

3 0 ~~31~~

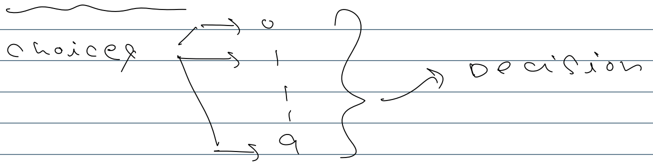
~~300~~

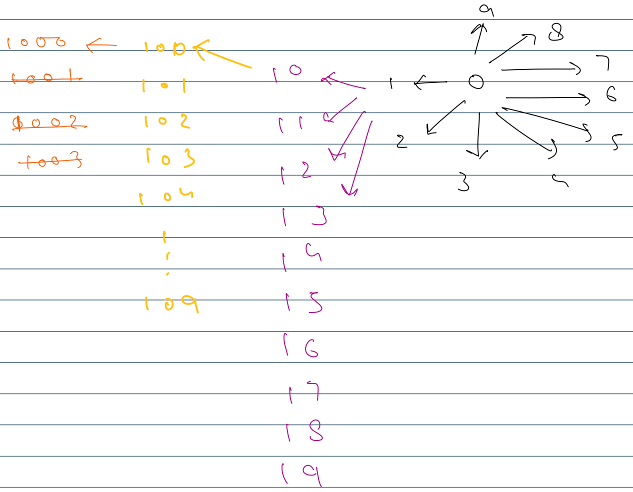
4 5 6 7 8 9

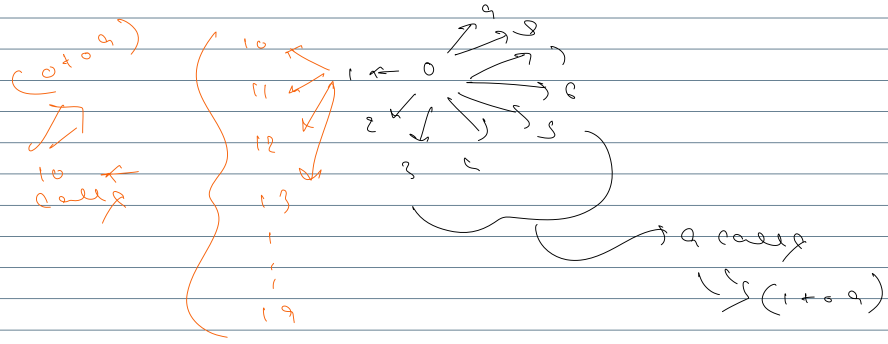
0 1 10 11 12 13 14 15 16 17 18 19 2 20 21 22

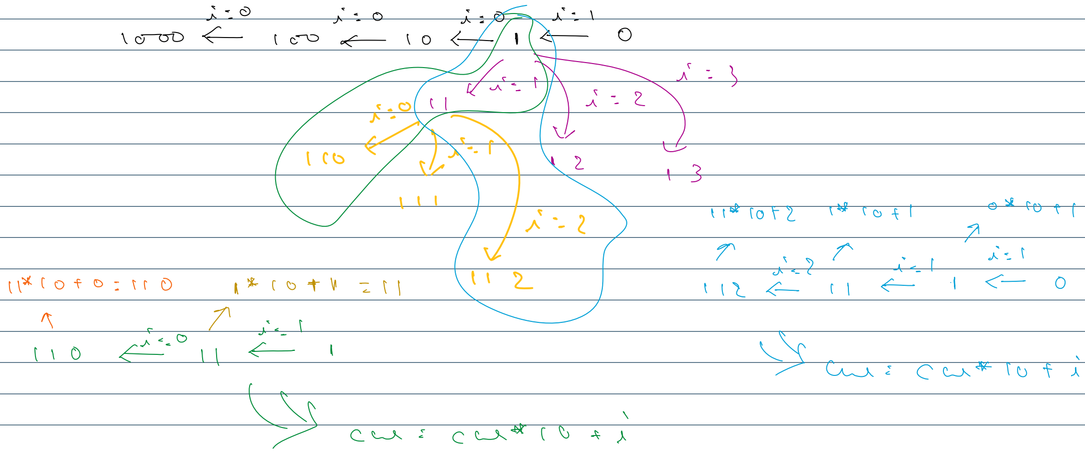
23 -- - - 29 3 30, 4, 5, 6, 7, 8, 9

Recursion

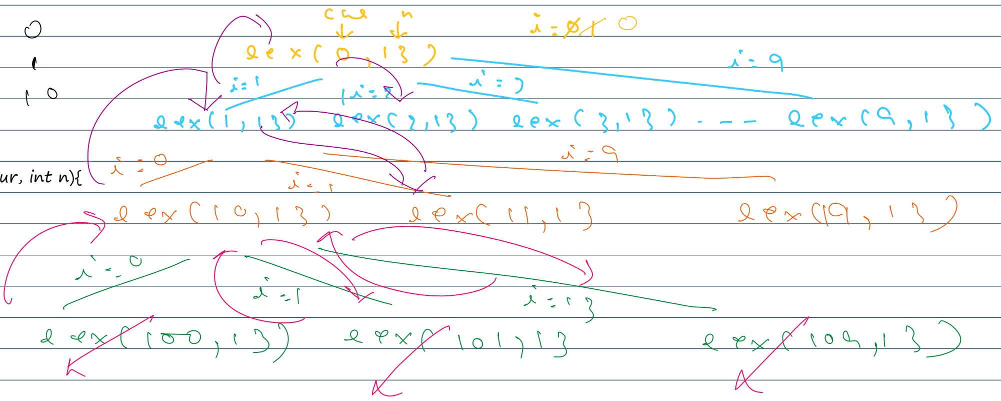


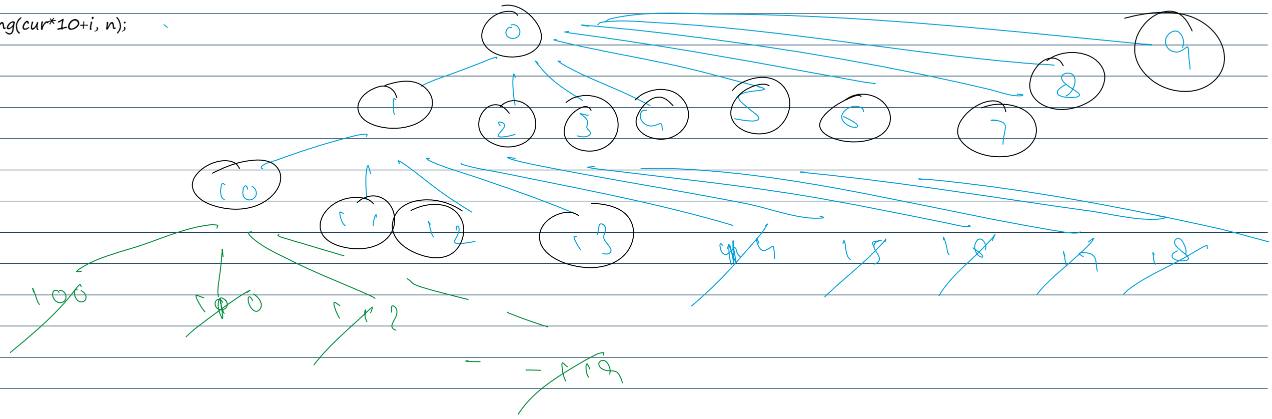




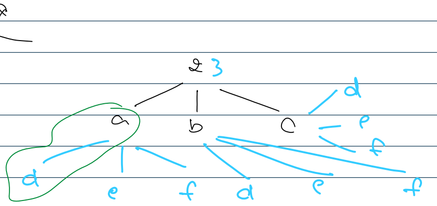


```
public class Main {
    public static void main(String[] args) {
        int n = ~~30~~; 13
        lexicographicalCounting(0, n);
    }

    public static void lexicographicalCounting(int cur, int n){
        if(cur > n){
            return;
        }
        System.out.print(cur+" ,");

        int i = 0;
        if(cur == 0){
            i = 1;
        }
        for(; i<=9; i++){
            lexicographicalCounting(cur*10+i, n);
        }
    }
}
```

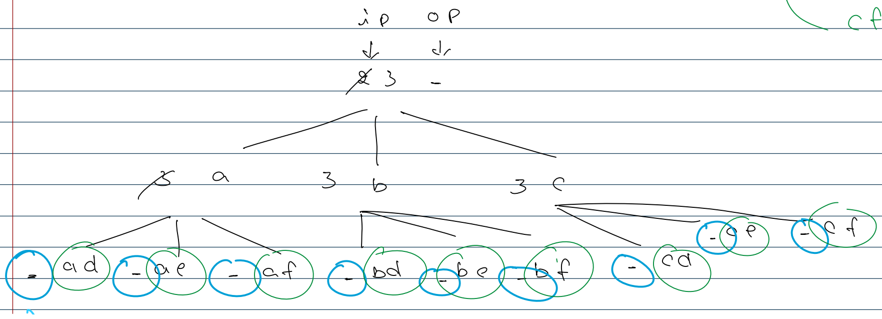

0
1
10
cur n
lex(0, 13)
i = 0 1 0
i = 9
i=1
i=2
i=3
lex(1,13) lex(2,13) lex(3,13) --- lex(9,13)
i=0
i=1
i=9
lex(10,13) lex(11,13) lex(19,13)
i=0
i=1
i=13
lex(100,13) lex(101,13) lex(109,13)



Letter Combinations

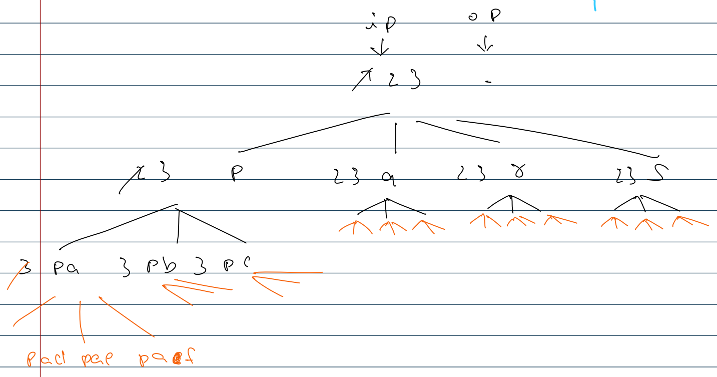


ad
ae
af
bd
be
bf
cd
ce
cf

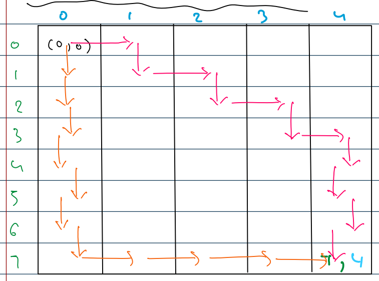


```java
class Solution {
    public List<String> letterCombinations(String digits) {
        String ip = digits;
        String op = "";
        List<String> list = new ArrayList<>();
        if(digits == null || digits.length() == 0){
            return list;
        }
        letterCombinationsUtil(ip, op, list);
        return list;
    }
    static String[] keys = {"", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"};
    public static void letterCombinationsUtil(String ip, String op, List<String> list){
        if(ip.length() == 0){
            // System.out.println(op);
            list.add(op);
            return;
        }
        String ch = ip.charAt(0)+"";
        String pressedKey = keys[Integer.valueOf(ch)];
        for(int i=0; i<pressedKey.length(); i++){
           letterCombinationsUtil(ip.substring(1), op + pressedKey.charAt(i), list);
        }
        // letterCombinationsUtil(ip.substring(1), op + pressedKey.charAt(0));
        // letterCombinationsUtil(ip.substring(1), op + pressedKey.charAt(1));
        // letterCombinationsUtil(ip.substring(1), op + pressedKey.charAt(2));
    }

}
```

Base cndtn ⇒ smallest valid ip ⇒ if (ip.length() == 0) {
return; ⇒ answer
}



Grid Path finder



move → right (vertically)
→ down (horizontally)

V H V H V H V H H H H H

H H H H H H H V V V V

H and V ⇒ should be swapped → omission